**Seminar on Big Data Management**

Hannu Kämäri

13.3.2017

# CLASH OF TITANS

# Motivation

- MapReduce and Spark are very popular frameworks.
- Apache Spark and HDFS has been under an investigation in NLS.
- Apache Hadoop is propably most famous BigData tool.
- Data processing is becoming more and more a central point in IT.

# Paper

## Clash of the Titans: MapReduce vs. Spark for Large Scale Data Analytics

Juwei Shi[‡], Yunjie Qiu[†], Umar Farooq Minhas[§], Limei Jiao[†], Chen Wang[♯], Berthold Reinwald[§], and Fatma Özcan[§] *
[†]IBM Research - China [§]IBM Almaden Research Center
[‡]DEKE, MOE and School of Information, Renmin University of China [♯]Tsinghua University

### ABSTRACT

MapReduce and Spark are two very popular open source cluster computing frameworks for large scale data analytics. These frameworks hide the complexity of task parallelism and fault-tolerance, by exposing a simple programming API to users. In this paper, we evaluate the major architectural components in MapReduce and Spark frameworks including: shuffle, execution model, and caching, by using a set of important analytic workloads. To conduct a detailed analysis, we developed two profiling tools: (1) We correlate the task execution plan with the resource utilization for both MapReduce and Spark, and visually present this correlation; (2) We provide a break-down of the task execution time for in-depth analysis. Through detailed experiments, we quantify the performance differences between MapReduce and Spark. Furthermore, we attribute these performance differences to different components which are architected differently in the two frameworks. We further expose the source of these performance differences by using a set of micro-benchmark experiments. Overall, our experiments show that Spark is about 2.5x, 5x, and 5x faster than MapReduce,

are: MapReduce [7] and Spark [19]. These systems provide simple APIs, and hide the complexity of parallel task execution and fault-tolerance from the user.

### 1.1 Cluster Computing Architectures

MapReduce is one of the earliest and best known commodity cluster frameworks. MapReduce follows the functional programming model [8], and performs explicit synchronization across computational stages. MapReduce exposes a simple programming API in terms of `map()` and `reduce()` functions. Apache Hadoop [1] is a widely used open source implementation of MapReduce.

The simplicity of MapReduce is attractive for users, but the framework has several limitations. Applications such as machine learning and graph analytics iteratively process the data, which means multiple rounds of computation are performed on the same data. In MapReduce, every job reads its input data, processes it, and then writes it back to HDFS. For the next job to consume the output of a previously run job, it has to repeat the read, process, and write cycle. For iterative algorithms, which want to read once, and iterate

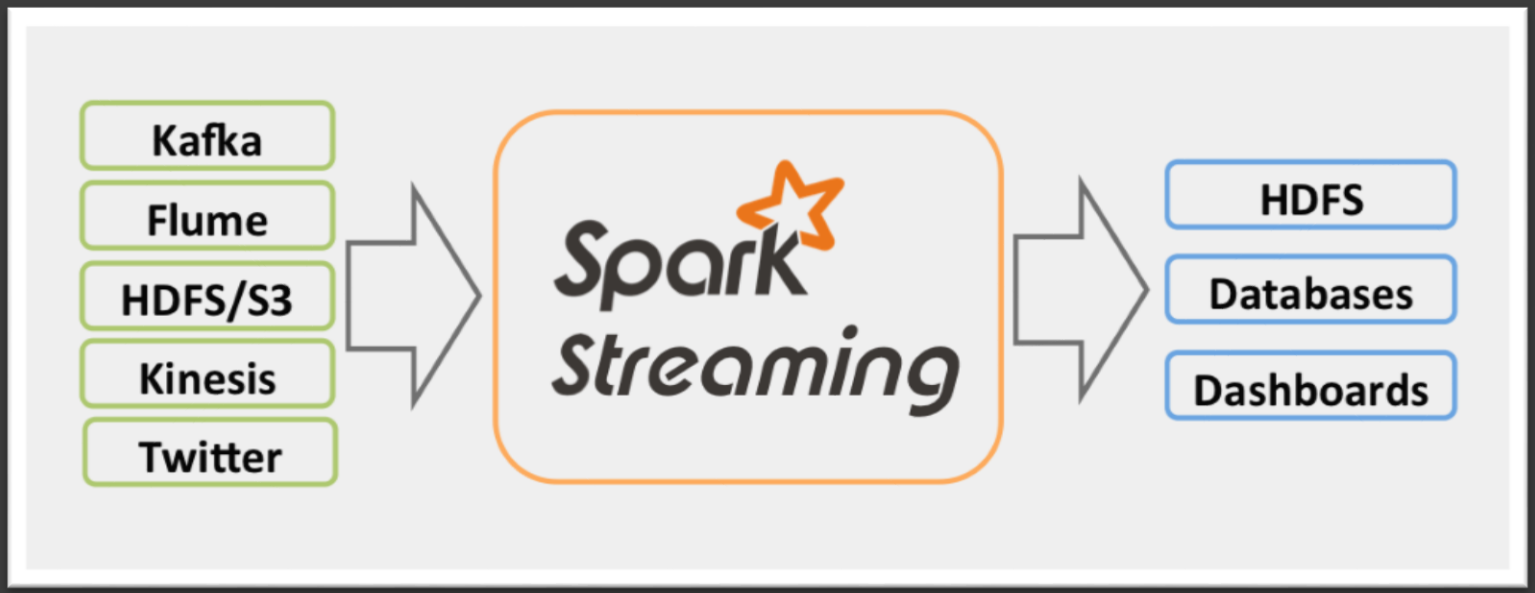
# Related work

- **Performance analysis of clustering algorithm under two kinds of big data architecture (Li & others 2017):** a theoretical and an experimental analysis on the two frameworks with k-means analysis. Their findings on both analysis showed that Spark is superior compared to MapReduce when comparing execution times or I/O.
- **An evaluation and analysis of graph processing frameworks on five key issues (Gao and others 2015):** message passing experiments with PageRank to compare Graph, Spark and MapReduce. Findings support that Spark is significantly faster than MapReduce.
- **Graysort competition (2014):** Spark was undoubtedly faster than MapReduce.

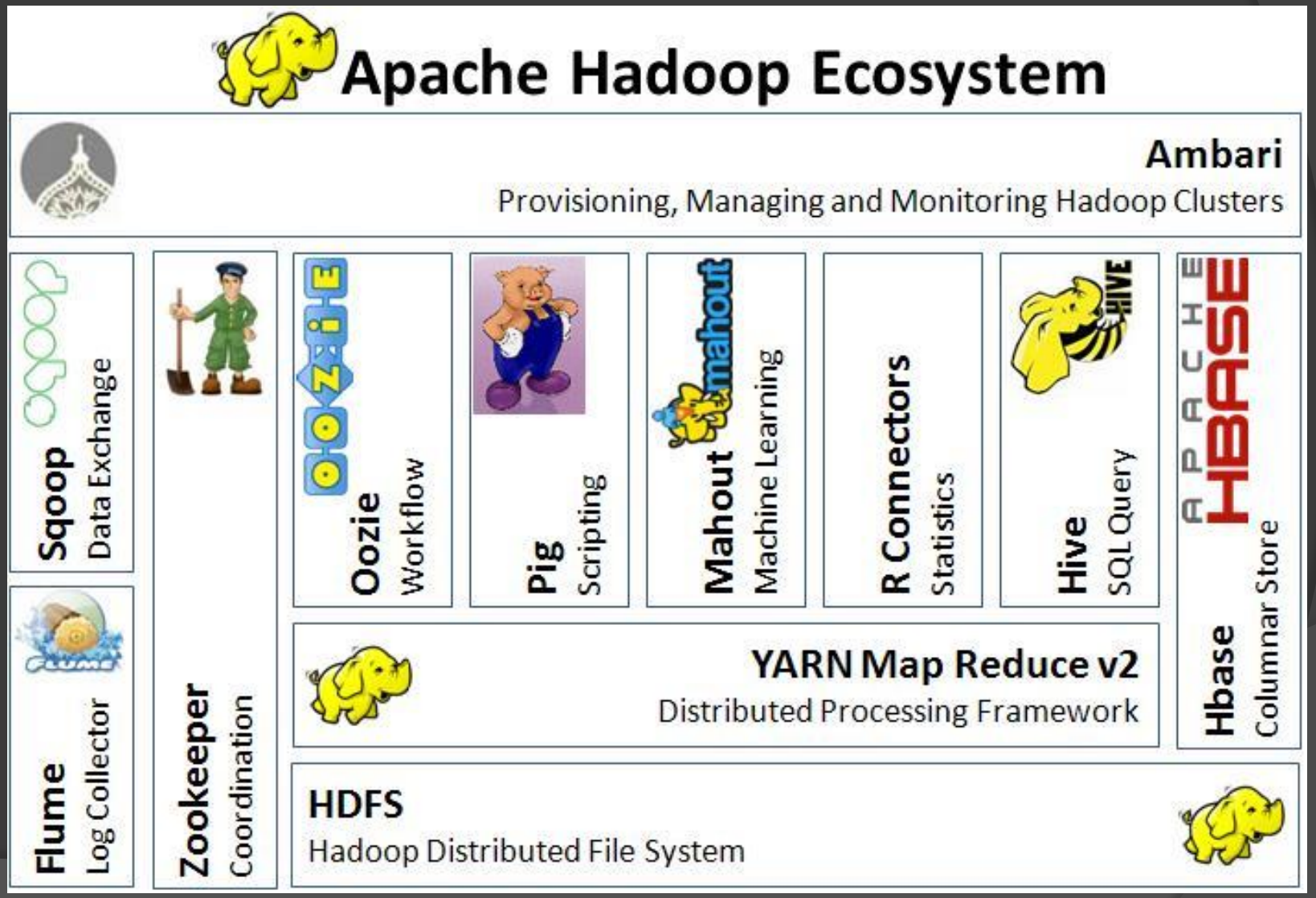
# Main algorithms

- Comparison was made between Spark and Hadoop.
- For both five workloads were run: Word Count, Sort, k-means, linear regression, and PageRank. Five tests were needed to adequately bring out differences between the two frameworks.
- For each job both one-pass and iterative options were evaluated.
- Tests were run with data that was suitable for each test and for bringing out differences between algorithms objectively.

# Spark


Kafka
Flume
HDFS/S3
Kinesis
Twitter
Spark
Streaming
HDFS
Databases
Dashboards

# Apache Hadoop



# Results

- To summarize:  Spark was faster than MapReduce in almost all tests.
- Major differences that cause Spark to be faster are RDD caching and better algorithms.
- Major (only) weak point in Spark compared to MapReduce exists in execution plans – in some cases MapReduce has better parallelism. This causes MapReduce to be faster in sort-tests with larger data sets.

# Why you should read the paper

- Easy to follow
- Very detailed
- Helps to deepen understanding about the frameworks
- Presentation and visualization of results

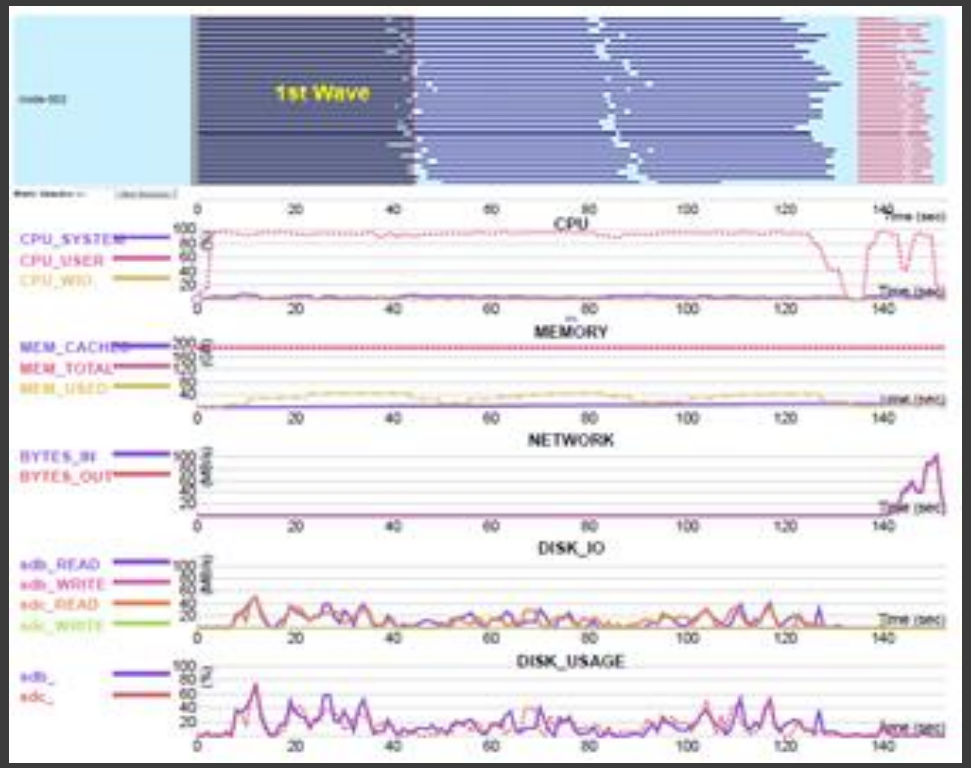
# Experimental setup

- Experimental setup is very well described. Both hardware and software are presented with all necessary details. Hardware details do not only include RAM, CPU, and memory details. Also exact details about disk bandwidth, Ethernet, and virtual clusters are introduced.
- Software configuration is made with equal carefulness.
- Same care for details exists also for workload descriptions.

# Depth and coverage of analysis

- Depth of analysis is excellent. Results are presented for all architectural elements and analysis include reasoning of results - what caused different performance and how different configurations impact the results.
- Configuration adjustments show outstanding understanding about execution details on both software and hardware levels.
- For example authors compare different types of cache options and explain thoroughly how each of them affects different stages of execution.
- Breakdown of results is done with great care and it helps readers to understand how the two algorithms really work.

# Visualization




| Platform | Spark | MR | Spark | MR | Spark | MR |
|---|---|---|---|---|---|---|
| Input size (GB) | 1 | 1 | 40 | 40 | 200 | 200 |
| Number of map tasks | 9 | 9 | 360 | 360 | 1800 | 1800 |
| Number of reduce tasks | 8 | 8 | 120 | 120 | 120 | 120 |
| Job time (Sec) | 30 | 64 | 70 | 180 | 232 | 630 |
| Median time of map tasks (Sec) | 6 | 34 | 9 | 40 | 9 | 40 |
| Median time of reduce tasks (Sec) | 4 | 4 | 8 | 15 | 33 | 50 |
| Map Output on disk (GB) | 0.03 | 0.015 | 1.15 | 0.7 | 5.8 | 3.5 |

# How the paper could have been improved

- Literature analysis is weak. Authors only conclude that no other papers could be found.
- Authors do not include any discussion about relevance of the analysis conducted.

# What really matters?

- Question is in what kind of environment these differences even matter?
- Does it matter if 100 GB of data can be sorted in three minutes instead of five or word count takes for 200 GB of data takes 33 seconds instead of 50?
- Most of these technologies have been created by large social media companies (Facebook, LinkedIn, etc.) and are being used to analyze their data.
- How many users are actually dealing with data sets in that scale with such time constraints?

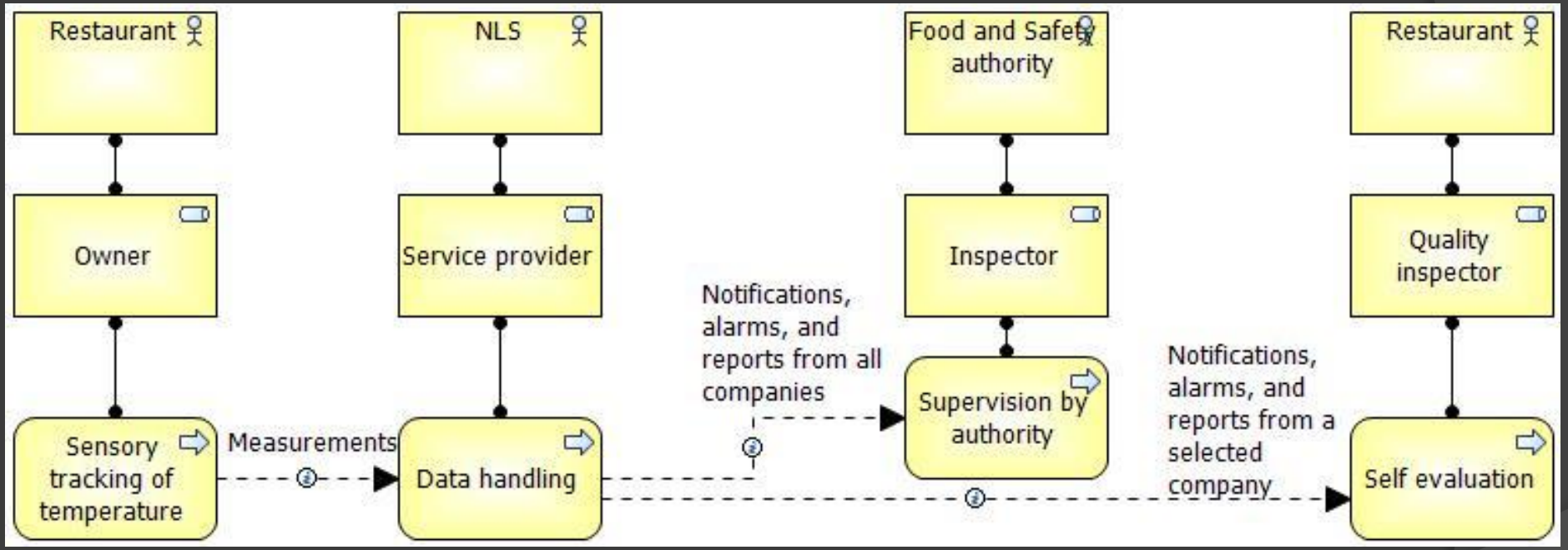
# Trends in BigData

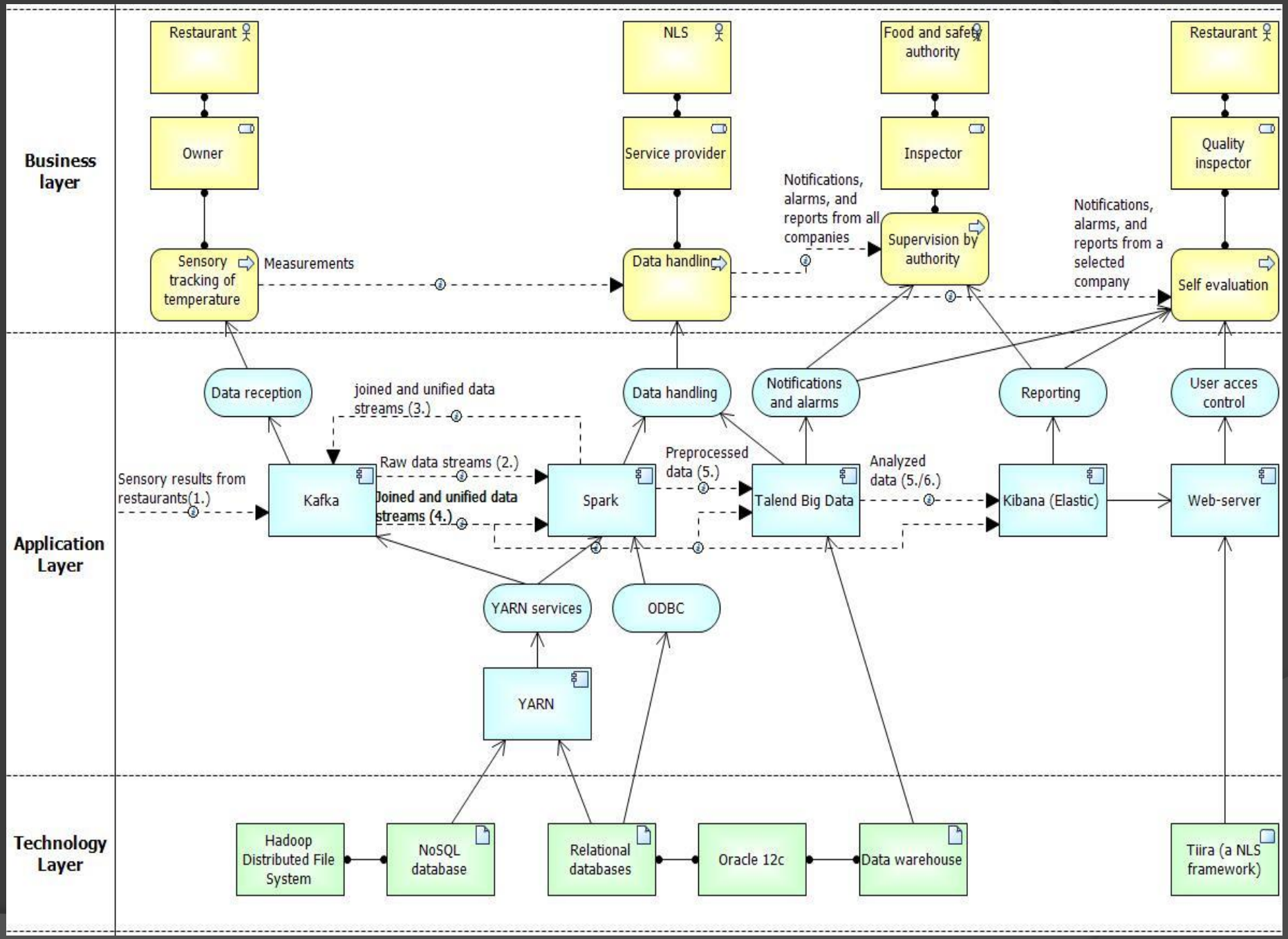
- I think the authors are missing a point.
- Big Data technologies are focusing more and more on stream processing.
- Apache project list contains no new entries for batch processing software.
- there are new tools for stream processing such as Spark, Samza, and Storm.
- Apache has also developed tool for distributed computing - Apache Edgent that can be run on micro-kernels. Kafka - a message broker designed to Big Data use.  Etc. Etc.

# Conclusions

- More vital discussion would handle primary use cases for each technology and how they should be combined. It is not an easy task.
- Apache alone has 37 seven tools for Big Data management.
- How they should be used and how easy they are to use would be a more interesting story.
- And then the analyze could focus on total throuput? In other words – use cases do matter.

# Own example


Restaurant
NLS
Food and Safety authority
Restaurant
Owner
Service provider
Inspector
Quality inspector
Notifications, alarms, and reports from all companies
Supervision by authority
Notifications, alarms, and reports from a selected company
Sensory tracking of temperature
Measurements
Data handling
Self evaluation


Business layer
Application Layer
Technology Layer
Restaurant
NLS
Food and safety authority
Restaurant
Owner
Service provider
Inspector
Quality inspector
Notifications, alarms, and reports from all companies
Notifications, alarms, and reports from a selected company
Sensory tracking of temperature
Measurements
Data handling
Supervision by authority
Self evaluation
Data reception
joined and unified data streams (3.)
Data handling
Notifications and alarms
Reporting
User acces control
Sensory results from restaurants(1.)
Kafka
Raw data streams (2.)
Joined and unified data streams (4.)
Spark
Preprocessed data (5.)
Talend Big Data
Analyzed data (5./6.)
Kibana (Elastic)
Web-server
YARN services
ODBC
YARN
Hadoop Distributed File System
NoSQL database
Relational databases
Oracle 12c
Data warehouse
Tiira (a NLS framework)